**Experiment No. 01**

**Aim:** Write a program to demonstrate basic syntactical constructs of C++.

1 Operators & Expressions

2. Looping Statement

3. Decision Making Statement

Erons formula

Relational and logical operators

**program**

**triangle type**

**matrix multiplication**

**switch case program any one of menu or bill**

**Theory:**

**Arithmetic Operators:**

[Arithmetic operators in C++](https://www.tutorialspoint.com/cplusplus/cpp_arithmatic_operators.htm) are the basic operators, which are used for basic mathematical or arithmetical operations on operands. These operators are essential for performing calculations and manipulating data within a program.

There are following arithmetic operators supported by C++ language −

Assume variable A holds 10 and variable B holds 20, then −

| **Operator** | **Description** | **Example** |
| --- | --- | --- |
| + | Adds two operands | A + B will give 30 |
| - | Subtracts second operand from the first | A - B will give -10 |
| \* | Multiplies both operands | A \* B will give 200 |
| / | Divides numerator by de-numerator | B / A will give 2 |
| % | Modulus Operator and remainder of after an integer division | B % A will give 0 |
| ++ | [**Increment operator**](https://www.tutorialspoint.com/cplusplus/cpp_increment_decrement_operators.htm), increases integer value by one | A++ will give 11 |
| -- | [**Decrement operator**](https://www.tutorialspoint.com/cplusplus/cpp_increment_decrement_operators.htm), decreases integer value by one | A-- will give 9 |

**Relational Operators**

[Relational operators](https://www.tutorialspoint.com/cplusplus/cpp_relational_operators.htm) are used to compare two values or expressions. These operators return a boolean value − **true** if the comparison is correct, and else **false**.

They are essential for making decisions and controlling the flow of a program based on conditions.

There are following relational operators supported by C++ language

Assume variable A holds 10 and variable B holds 20, then −

| **Operator** | **Description** | **Example** |
| --- | --- | --- |
| == | Checks if the values of two operands are equal or not, if yes then condition becomes true. | (A == B) is not true. |
| != | Checks if the values of two operands are equal or not, if values are not equal then condition becomes true. | (A != B) is true. |
| > | Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true. | (A > B) is not true. |
| < | Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true. | (A < B) is true. |
| >= | Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true. | (A >= B) is not true. |
| <= | Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true. | (A <= B) is true. |

The if….else Statement

Use the else statement to specify a block of code to be executed if the condition is false.

**Syntax:**

if (*condition*) {

*// block of code to be executed if the condition is true*

}

else {

*// block of code to be executed if the condition is false*

}

**Looping in C++:**

**For Loop**

When you know exactly how many times you want to loop through a block of code, use the for loop instead of a while loop:

**Syntax:**

for (*statement 1*; *statement 2*; *statement 3*) {

*// code block to be executed*

}

Statement 1 is executed (one time) before the execution of the code block.

Statement 2 defines the condition for executing the code block.

Statement 3 is executed (every time) after the code block has been executed.

**While Loop:**

The while loop loops through a block of code as long as a specified condition is true:

Syntax:

while (*condition*) {

*// code block to be executed*

}

**The Do/While Loop:**

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax:

do {

*// code block to be executed*

}

while (*condition*);

**Conclusion:** Students are able to learn operators, statement and various loops of c++.

**Experiment No. 02**

**Aim:** Write a program to demonstrate decision making statements and control structure, Arrays in C++.

**Theory:**

**Switch statement in C++:**

Syntax:

switch (expression) {

case constant1:

// code to be executed if expression equals constant1

break;

case constant2:

// code to be executed if expression equals constant2

break;

// You can have any number of case statements

default:

// code to be executed if none of the cases match

Key Points

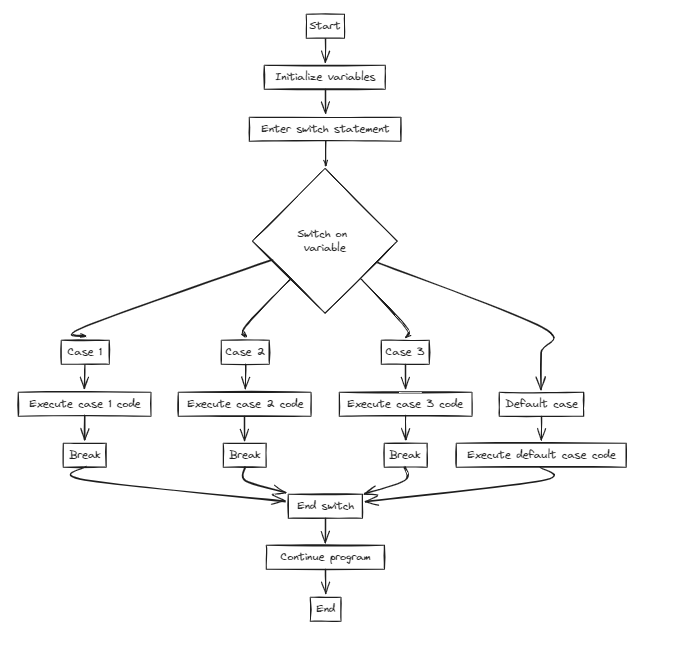
**Expression:** The value being evaluated in the switch statement. It must be of an integral or enumeration type.

**Case Labels:** Each case label must be a constant expression. These labels are compared against the value of the expression.

**Break Statement:** The break statement prevents the execution from falling through to subsequent cases. Without it, all subsequent cases are executed regardless of their labels.

**Default Case:** The default case is optional but recommended. It provides a fallback when none of the specified cases match the expression.

Flowchart of Switch Statement in C++



**C++ Arrays:**

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

To declare an array, define the variable type, specify the name of the array followed by square brackets and specify the number of elements it should store:

string cars[4];

We have now declared a variable that holds an array of four strings. To insert values to it, we can use an array literal - place the values in a comma-separated list, inside curly braces:

string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};

To create an array of three integers, you could write:

int myNum[3] = {10, 20, 30};

Access the Elements of an Array

You access an array element by referring to the index number inside square brackets [].

This statement accesses the value of the first element in cars:

Example

string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};

cout << cars[0];

// Outputs Volvo

**Conclusion:** Students are able to understand switch and working of array in c++ programming.